Internet-Draft
<draft-satran-scot-02.txt>
Expires 14 August 2000

J. Satran
D. Smith
K. Meth
IBM
C. Sapuntzakis
Cisco Systems
M. Toledano
P. Sarkar
C. Fuente
IBM
E. Zeidner
SanGate

February 2000

SCSI/TCP (SCSI over TCP)

Status of this Memo

This document is an Internet-Draft and is in full conformance with all provisions of Section 10 of RFC2026.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt

The list of Internat-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html.

Exhibit B

Table of Contents

1. Abstract

The Small Computer Systems Interface (SCSI) is a popular family of protocols for communicating with I/O devices, especially storage devices.

This memo describes a transport protocol for SCSI that operates on top of TCP.

The SCSI/TCP protocol aims to be fully compliant with the requirements laid out in the SCSI Architecture Model - 2 [SAM2] document.

2. Overview

2.1. SCSI Concepts

The endpoint of most SCSI commands is a "logical unit" (LUN). Examples of logical units include hard drives, tape drives, CD and DVD drives, even printers and processors.

A "target" is a collection of logical units and is directly addressable on the network. The target corresponds to the server in the client-server model.

An "initiator" creates and sends SCSI commands to the target. The initiator corresponds to the client in the client-server model.

A "task" is a linked set of SCSI commands. Some LUNs support multiple simultaneous tasks. The target uses a "task tag" to distinguish between simultaneous tasks. Only one command in a task can be outstanding at any given time.

A SCSI command results in a data phase and a response phase. In the data phase, information travels either from the initiator to the target, as in a WRITE command, or from target to initiator, as in a READ command. In the response phase, the target returns the final status of the operation, including any errors. A response terminates a SCSI command.

2.2. SCSI/TCP Functional Overview

Communication between initiator and target occurs over one or more TCP connections. The first TCP connection opened is designated a control connection and used for sending control messages, SCSI commands, and parameters. Additional connections may be opened for sending data from the SCSI data phases.

2.3. SCSI/TCP Login

The purpose of SCSI/TCP login is to create a connection, authenticate the parties, and authorize the initiator to send SCSI commands.

The targets listen on a well-known TCP port for incoming connections. The initiator begins the login process by connecting to that well-known TCP port.

As part of the login process, the initiator and target MAY wish to authenticate each other. This can occur in many different ways. For example, the endpoints may wish to check the IP address of the other party. If the TCP connection uses transport layer security [TLS], certificates may be used to identify the endpoints. Also, SCSI/TCP includes commands for identifying the initiator and passing an authenticator to the target (see Appendix B).

Once suitable authentication has occured, the target MAY authorize the initiator to send SCSI commands. How the target chooses to authorize an initiator is beyond the scope of this document.

The target indicates a succesful authentication and authorization by sending a login response with "accept login".

After authentication and authorization, other parameters may be negotiated using the highly extensible Text Command message that allows arbitrary key:value pairs to be passed.

Finally, if any other TCP control or data connections between the initiator and target are currently open, they will be forced closed (TCP RST), flushing unacknowledged data.

2.4. SCSI/TCP Full Feature Phase

Once the initiator is authorized to do so, the connection is in SCSI/TCP full feature phase. The initiator may send SCSI commands to the various LUNs on the target.

SCSI commands are encapsulated in messages that go over the control connection.

Data phases associated with SCSI commands go over separate data connections. Initiators may explicitly request the establishment of data connections to targets using the "Open Data Connections" message. A target responds by granting some number of data connections, (to be established using the well known SCSI/TCP data port), and by providing a cookie for the initiator to produce upon establishment of its data connections.

The targets listen on another well-known TCP port for incoming SCSI/TCP data connections. The initiator connects to the well-known SCSI/TCP data connection port and provides the cookie it received in the "Open Data Connections" response. The cookie occupies the first 8 bytes of data sent by the initiator through the data connection. The target uses the cookie to match a newly established data channel with its corresponding control channel.

2.5. SCSI/TCP Connection Termination

Graceful connection shutdowns are done by sending TCP FINs. Graceful connection shutdowns MUST only occur when there are no outstanding tasks on the connection. A target SHOULD respond rapidly to a FIN from the initiator by closing its half of the connection.

Usually, the initiator will initiate the closing of data channels when it no longer needs them for its data transfer operations. Similarly, an initiator may initiate the closing of its control channel when it has finished all operations with the target device.

The closing of one data channel has no effect on other data channels connecting the initiator and the target.

A target may wish to close a TCP data connection. Once an initiator has received the FIN, it SHOULD not add any more data to be sent onto that connection and should close its half of the connection when it is done sending the pending data.

In the case where a control channel is closed, the target should clean up all of its state associated with the corresponding initiator; all outstanding tasks are cancelled and all resources that were allocated for the initiator can be freed. Any open data connections should be forcibly closed (using TCP RST).

2.6. Naming

Targets are named using a combination of a domain name and an optional path. The leftmost slash (/) character separates the domain name from the path. The initiator uses the domain name to connect to a remote network endpoint. The path is uninterpreted by the initiator.

Examples:

Target:tapefarm.acme.com/tape37
Target:bigtape.acme.com
Target:sample.acme.com/paths/can/have/many/slashes/and/d.o.t.s./

Initiators are named using a combination of a domain name and an optional principal. The principal is separated from the domain name by the rightmost @ sign.

Examples:

Initiator:fred@machine32.customer.com
Initiator:machine32.customer.com

A domain name that contains exactly four numbers separated by dots (.), where each number is in the range 0 through 255, will be interpreted as an IPv4 address. For example:

Target:10.0.0.1/tape43

3. Message Formats

All multi-byte integers specified in formats defined in this document are to be represented in network byte order (i.e., big endian).

3.1. Template Header

All SCSI/TCP messages and responses have a header of the same length (consisting of 40 bytes). Additional data may be added, as necessary, beginning with byte 40. The fields of Opcode and Length appear in all message and response headers. The other most commonly used fields are Initiator Task Tag, Logical Unit Number, and Flags, which, when used, always appear in the same location of the header.

```
Byte /    0      |      1      |      2      |      3      |
 /               |             |             |             |
 |7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|
 +---------------+---------------+---------------+---------------+
 0| Opcode       | Reserved (0)  | Length of Data After Header   |
 +---------------+---------------+---------------+---------------+
 4| Initiator Task Tag                                           |
 +---------------+---------------+---------------+---------------+
 8| Logical Unit Number (LUN)                                    |
 +                                                               +
12|                                                              |
 +---------------+---------------+---------------+---------------+
16| Flags                        | Reserved (0)                  |
 +---------------+---------------+---------------+---------------+
20/ Opcode-specific data                                         /
 +/                                                              /
 +---------------+---------------+---------------+---------------+
40
```

3.1.1. Opcode

The Opcode indicates which SCSI/TCP type of message or response is encapsulated by the header. Valid opcodes for messages (sent by initiator to target) are:

0x00 Ping Command (from initiator to target)
0x01 SCSI Command (encapsulates a SCSI Command Block)
0x02 SCSI Task Management Message
0x03 Login Command
0x04 Open Data Connections Command
0x05 Text Command

Valid opcodes for responses (sent by target to initiator) are:

- 0x80 Ping Response (from target to initiator)
- 0x81 SCSI Response (contains SCSI status and possibly sense informaton or other response information)
- 0x82 SCSI Task Management Response
- 0x83 Login Response
- 0x84 Open Data Connections Response
- 0x85 Text Response
- 0x86 Ready To Transfer (RTT - sent by target to initiator when it is ready to receive data from initiator)
- 0x87 Asynchronous Event (sent by target to initiator to indicate certain special conditions)
- 0x88 Opcode Not Understood

3.1.2. Length

The Length field indicates the number of bytes, beyond the 40 byte header, that are being sent together with this message header. It is anticipated that most SCSI/TCP messages and responses (not counting data transfer messages) will not need more than the 40 byte header, and hence the Length field will contain the value 0. All messages using this header are sent over the control channel. Bulk data transfers are performed over the data channels, and have a different header format.

Large amounts of data accompanying a SCSI command should not be sent over the control channel. If too much data is sent over the control channel, there is the possibility that it would fill the TCP window, thus preventing other critical control commands from passing through the control channel. The control channel should be kept open at all times so that important messages can always pass through. (These may include a message to cancel a data transfer or to reset a device that has somehow entered a bad state). The Length field is, therefore, intentionally limited to 16 bits, allowing only up to 64KB of data to be passed with any particular control message. Any large data transfers for READ and WRITE operations (including unsolicited WRITE operations) should be sent over the data channels.

3.1.3. LUN

The LUN specifies the Logical Unit for which the command is targeted. If the command does not relate to a Logical Unit, this field is either ignored or may be used for some other purpose. According to [SAM2], a Logical Unit Number can take up to a 64-bit field that identifies the Logical Unit within a target device. The exact format of this field can be found in the [SAM2] document.

3.1.4. Initiator Task Tag

The initiator assigns a Task Id (or tag) to each SCSI task that it issues. This Tag is a initiator-wide unique identifier that can be used to uniquely identify the Task.

3.1.5. Flags and additional information

These field have different meanings for different messages.

3.2. SCSI Command

```
Byte /  0    |    1    |    2    |    3    |
 /        |        |        |        |
|7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|
 +---------------+---------------+---------------+---------------+
 0| Opcode (0x01) | Reserved (0)  | Length                      |
 +---------------+---------------+---------------+---------------+
 4| Initiator Task Tag                              |
 +---------------+---------------+---------------+---------------+
 8| Logical Unit Number (LUN)                                 |
 +                                              +
12|                                             |
 +---------------+---------------+---------------+---------------+
16|Rsrvd (0)|A|R|W|Rsrved(0)|ATTR | Reserved (0)                 |
 +---------------+---------------+---------------+---------------+
20| Expected Data Transfer Length                        |
 +---------------+---------------+---------------+---------------+
24| SCSI Command Block (SCB)                               |
 +                                              +
28|                                             |
 +                                              +
32|                                             |
 +                                              +
36|                                             |
 +---------------+---------------+---------------+---------------+
40/ Additional Data (Command Dependent)                        /
 +/                                            /
 +---------------+---------------+---------------+---------------+
```

3.2.1. Flags

The Flags field for a SCSI Command consists of two bytes. (In general, one byte is used for data flow information while the other byte is used for Task Attributes information.)

```
        Byte 16
    b0  (W) set when data is expected to flow from initiator to
        target (write).
    b1  (R) set when data is expected to flow from target to ini-
        tiator (read).
    b2  (A) set to turn off Autosense for this command (see
        [SAM2]).
    b3-7 not used (should be set to 0).
```

Autosense refers to the automatic return of sense data to the initiator in case a command did not complete successfully. If autosense is turned off, the initiator must explicitly request that sense data be sent to it after some command has completed with a CHECK CONDITION status.

Byte 17
b0-2 used to indicate Task Attributes.
b3-7 not used (should be set to 0).

3.2.2. Task Attributes

The Task Attribute field (ATTR) can have one of the following integer values (see [SAM2] for details):

0	Untagged
1	Simple
2	Ordered
3	Head of Queue
4	ACA

3.2.3. Expected Data Transfer Length

The Expected Data Transfer Length field states the number of bytes expected to be sent over the data channel for this SCSI operation. (The SCSI command itself is sent over the control channel.)

For a WRITE operation, the initiator uses this field to specify the number of bytes of data it expects to transfer for this operation over the data channel(s) (not counting data headers).

For a READ operation, the initiator uses this field to specify the number of bytes of data it expects the target to transfer to the initiator over the data channel(s).

If no data will be transferred over the data channels for this SCSI operation, this field should be set to 0.

If data is sent together with the SCSI command over the control channel, the byte count should be included in the Length field (bytes 2-3).

The target knows to expect data for this SCSI command over the data channel by receiving a non-zero value in the Expected Data Transfer Length field.

Note that large amounts of data should be sent exclusively over the data channels so as not to clog the control channel. It is desired that the control channel be available at all times in order to be able to send critical messages (such as to interrupt the current data transfer). Upon completion of a data transfer, the target will inform the initiator of how many bytes were actually processed (sent or received) by the target.

3.2.4. SCSI Command Block (SCB)

There are 16 bytes in the SCB field, designed to accomodate the largest currently defined SCB.

If, in the future, larger SCB's are allowed, the spill-over of the SCB may extend beyond the 40-byte boundary, followed by the data or parameters for the SCB. The target will use the SCSI/TCP Length field plus the information in the SCB to figure out how many additional bytes are part of the SCB, with the remaining bytes serving as the data and parameters of the SCB.

3.2.5. Command Data

Some SCSI commands require additional parameters or data to accompany the SCSI command. This data may be placed beyond the 40-byte boundary of the SCSI/TCP header. The Length field is set to the length of this data beyond the 40-byte header. Note that the Length field is intentionally limited to 2 bytes, thus limiting the amount of data to 64K.

Any SCSI command whose data/parameters require more than 64K beyond the 16 byte SCB must utilize one of the data connections to perform the data transfer.

3.3. SCSI Response

```
 Byte /   0     |     1     |     2     |     3     |
   /          |           |           |           |
   |7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|
   +---------------+---------------+---------------+---------------+
  0| Opcode (0x81) | Reserved (0)  | Length                        |
   +---------------+---------------+---------------+---------------+
  4| Initiator Task Tag                                          |
   +---------------+---------------+---------------+---------------+
  8| Logical Unit Number (LUN)                                     |
   +                                                               +
 12|                                                               |
   +---------------+---------------+---------------+---------------+
 16| Rsrved(0) |O|U| Reserved (0)  | Reserved (0)                  |
   +---------------+---------------+---------------+---------------+
 20| Residual Count                                                |
   +---------------+---------------+---------------+---------------+
 24| Command Status|SCSI/TCP Status| Reserved (0)                  |
   +---------------+---------------+---------------+---------------+
 28/ Reserved (0)                                                  /
  +/                                                               /
   +---------------+---------------+---------------+---------------+
 40/ Response or Sense Data (optional)                             /
  +/                                                               /
   +---------------+---------------+---------------+---------------+
```

3.3.1. Flags

The SCSI Response has its own set of flags, that differs from the flags for a SCSI Command.

Byte 16

b0 (U) set for Residual Underflow. In this case, the Residual Count indicates how many bytes were not transferred out of those expected to be transferred.

b1 (O) set for Residual Overflow. In this case, the Residual Count indicates how many bytes could not be transferred because the initiator's Expected Data Transfer Length was too small.

b2-7 not used (should be set to 0).

Bits 0 and 1 are mutually exclusive.

3.3.2. Residual Count

The Residual Count field is valid only in case either the Residual Underflow bit or Residual Overflow bit is set. If neither bit is set, the Residual Count field will be 0.

If the Residual Underflow bit is set, the Residual Count indicates how many bytes were not transferred out of those expected to be transferred.

If the Residual Overflow bit is set, the Residual Count indicates how many bytes could not be transferred because the initiator's Expected Data Transfer Length was too small.

3.3.3. Command Status

The Command Status field is used to report the SCSI status of the command (as specified in [SAM2]).

3.3.4. SCSI/TCP Status

The SCSI/TCP Status field is used to report the status of the command before it was sent to the LUN. The values are given below.

1 Non-existant LUN

3.3.5. Response or Sense Data

If Autosense was not disabled in the originating SCB and the Command Status was CHECK CONDITION (0x02), then the response field will contain sense data for the failed command.

If the Command Status is Good (0x00) and there are no data streams opened, then the Response Data field will contain data from the data phase of the SCB.

The Length parameter specifies the number of bytes in this field. If no error occured, and no data is needed for the response to the SCSI Command the Length field is 0.

Note that if the Command Status was CHECK CONDITION but Autosense was disabled, then sense data must be explicitly requested by the initiator with a new SCSI command.

3.4. Asynchronous Event

An Asynchronous Event may be sent from the target to the initiator without corresponding to a particular command. The target specifies the status for the event and sense data.

```
Byte /    0     |     1     |     2     |     3     |
   /            |           |           |           |
  |7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|
  +---------------+---------------+---------------+---------------+
 0| Opcode (0x87) | Reserved (0)  | Length                        |
  +---------------+---------------+---------------+---------------+
 4| Reserved (0)                                                  |
  +---------------+---------------+---------------+---------------+
 8| Logical Unit Number (LUN)                                     |
  +                                                               +
12|                                                               |
  +---------------+---------------+---------------+---------------+
16| Reserved (0)                                                  |
  +                                                               +
20|                                                               |
  +---------------+---------------+---------------+---------------+
24| Command Status|SCSI/TCP Status| Reserved (0)                  |
  +---------------+---------------+---------------+---------------+
28|Event Indicator| Reserved (0)                                  |
  +---------------+---------------+---------------+---------------+
32| Reserved (0)                                                  |
  +                                                               +
36|                                                               |
  +---------------+---------------+---------------+---------------+
40/ Sense Data                                                    /
 +/                                                               /
  +---------------+---------------+---------------+---------------+
```

3.4.1. SCSI/TCP Status

Some Asynchronous Events are strictly related to SCSI/TCP while others are related to SAM-2. The codes returned for SCSI/TCP Asynchronous Events are:

2 Target is being reset.
3 Expired cookie was used to establish a data connection.

3.4.2. Event Indicator

The following values are defined. (See [SAM2] for details.)

1 An error condition was encountered after command completion.
2 A newly initialized device is available.
3 Some other type of unit attention condition has occured.
4 An asynchronous event has occured.

Sense Data accompanying the report identifies the condition. The Length parameter is set to the length of the Sense Data.

3.5. SCSI Task Management Message

```
Byte /   0     |     1     |     2     |     3     |
   /           |           |           |           |
  |7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|
  +---------------+---------------+---------------+---------------+
 0| Opcode (0x02) | Reserved (0) |      Length                   |
  +---------------+---------------+---------------+---------------+
 4| Initiator Task Tag                                            |
  +---------------+---------------+---------------+---------------+
 8| Logical Unit Number (LUN)                                     |
  +                                                               +
12|                                                               |
  +---------------+---------------+---------------+---------------+
16| Msg indicator | Reserved (0)                                  |
  +---------------+---------------+---------------+---------------+
20/ Reserved (0)                                                  /
 +/                                                               /
  +---------------+---------------+---------------+---------------+
40
```

3.5.1. Msg Indicator

The Task Management functions provide an initiator with a way to explicitly control the execution of one or more Tasks. The Task Management functions are summarized as follows (for a more detailed description see the [SAM2] document):

1 Abort Task---aborts the task identified by the Task Tag field.
2 Abort Task Set---aborts all Tasks issued by this initiator on the Logical Unit.
3 Clear ACA---clears the Auto Contingent Allegiance condition.
4 Clear Task Set---Aborts all Tasks (from all initiators) for the Logical Unit.
5 Logical Unit Reset.
6 Target Reset.

For the functions above except <Target Reset>, a SCSI Task Management Response is returned, using the Initiator Task Tag to identify the operation for which it is responding.

For the <Target Reset> function, the target cancels all pending operations. The target may send an Asynchronous Event to all attached initiators notifying them that the target is being reset. The target then closes all of its TCP connections.

3.6. SCSI Task Management Response

```
 Byte /    0      |      1      |      2      |      3      |
    /             |             |             |             |
   |7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|
   +---------------+---------------+---------------+---------------+
  0| Opcode (0x82) | Reserved (0)  | Length                        |
   +---------------+---------------+---------------+---------------+
  4| Initiator Task Tag                                            |
   +---------------+---------------+---------------+---------------+
  8| Logical Unit Number (LUN)                                     |
   +                                                               +
 12|                                                               |
   +---------------+---------------+---------------+---------------+
 16| Msg indicator | Reserved (0)  | Reserved (0)                  |
   +---------------+---------------+---------------+---------------+
 20| Reserved (0)                                                  |
   +---------------+---------------+---------------+---------------+
 24| Response      | Reserved (0)                                  |
   +---------------+---------------+---------------+---------------+
 28/ Reserved (0)                                                  /
  +/                                                               /
   +---------------+---------------+---------------+---------------+
 40
```

For the functions <Abort Task, Abort Task Set, Clear ACA, Clear Task Set, Logical Unit reset>, the target performs the requested Task Management function and sends a SCSI Task Management Response back to the initiator. The target includes all of the information the initiator provided in the SCSI Task Management Message, so the initiator can know exactly which SCSI Task Management Message was serviced. In addition, the target provides a Response indication which may take on the following values:

0 Function Complete
1 Function Rejected

For the <Target Reset> function, the target cancels all pending operations. The target may send an Asynchronous Event to all attached initiators notifying them that the target is being reset. The target then closes all of its TCP connections.

3.7. Ready To Transfer (RTT)

When an initiator has submitted a SCSI Command with data passing from the initiator to the target (write), the target may specify which blocks of data it is ready to receive. In general, the target may request the data blocks be delivered in whatever order is convenient for the target at that particular instant. This information is passed from the target to the initiator in the Ready To Transfer (RTT) message.

In order to allow write operations without RTT, the initiator and target must have agreed to do so by both sending the AllowNoRTT:yes key-pair attribute to each other (either during Login or through the Text Command/Response mechanism).

```
Byte /      0       |       1       |       2       |       3       |
    /               |               |               |               |
   |7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|
   +---------------+---------------+---------------+---------------+
  0| Opcode (0x86) | Reserved (0)  | Length                        |
   +---------------+---------------+---------------+---------------+
  4| Initiator Task Tag                                            |
   +---------------+---------------+---------------+---------------+
  8| Reserved (0)                                                  |
   +                                                               +
 12|                                                               |
   +                                                               +
 16|                                                               |
   +---------------+---------------+---------------+---------------+
 20| Expected Data Transfer Length                                 |
   +---------------+---------------+---------------+---------------+
 24| Data Offset                                                   |
   +---------------+---------------+---------------+---------------+
 28| Target Transfer Tag                                           |
   +---------------+---------------+---------------+---------------+
 32| Reserved (0)                                                  |
   +                                                               +
 36|                                                               |
   +---------------+---------------+---------------+---------------+
 40
```

3.7.1. Expected Data Transfer Length and Data Offset

The target specifies how many bytes it wants the initiator to send as a result of this RTT message. The target may request the data from the initiator in several chunks, not necessarily in the original order of the data. The target, therefore, also specifies a

Data Offset indicating the point at which the data transfer should begin.

3.7.2. Target Transfer Tag

The target assigns its own tag to each RTT request that it sends to the initiator. This can be used by the target to identify data it receives, and can also be used as an RDMA tag [RDMA].

3.8. SCSI Data

The initiator and target send data in messages over data channel(s). The typical data transfer specifies the length of the data payload, the Transfer Tag provided by the receiver for this data transfer, and a buffer offset. After sending the data for a particular SCSI command through a data channel, an end-of-data indication must be sent in that data channel. The end-of-data indication consists of a SCSI Data header specifying a zero length data payload. Note that the end-of-data indication must be sent in each data channel over which data was sent for the particular SCSI command. The end-of-data marker indicates that no more data for this command will pass through this data channel. The end-of-data indication SCSI Data header sent from a target to an initiator may optionally also contain the Command Status for the data transfer. In this case Sense Data cannot be sent together with the Command Status. The typical SCSI Data message has the following format:

```
Byte /     0       |       1       |       2       |       3       |
    /              |               |               |               |
   |7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|
   +---------------+---------------+---------------+---------------+
  0| Length                                                        |
   +---------------+---------------+---------------+---------------+
  4| Transfer Tag                                                  |
   +---------------+---------------+---------------+---------------+
  8| Buffer Offset                                                 |
   +---------------+---------------+---------------+---------------+
 12. Payload                                                       .
   +.                                                              .
   .                                                               .
   +---------------+---------------+---------------+---------------+
```

The end-of-data indication has the following format:

```
Byte /     0       |       1       |       2       |       3       |
    /              |               |               |               |
   |7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|
   +---------------+---------------+---------------+---------------+
  0| Length (0)                                                    |
   +---------------+---------------+---------------+---------------+
  4| Transfer Tag                                                  |
   +---------------+---------------+---------------+---------------+
  8| Command Status| Reserved (0)  | Rsvrd (0) |D|C| Reserved (0)  |
   +---------------+---------------+---------------+---------------+
```

3.8.1. Length

The length field specifies the total number of bytes in the following payload.

3.8.2. Transfer Tag

The Transfer Tag identifies the operation to which this data transfer belongs.

When the transfer is from the target to the initiator, the Transfer Task Tag is the Initiator Task Tag that was sent with the SCSI command.

When the transfer is from the initiator to the target, the Transfer Task Tag is the Target Transfer Tag when RTT is enabled, or the Initiator Task Tag when RTT is disabled.

3.8.3. Buffer Offset

The Buffer Offset field contains the offset of the following data against the complete data transfer. If the data transfer is not split over multiple Data transmissions, then this field will be zero.

3.8.4. Status Flags

The Status Flags field indicates how to interpret the Command Status field, when the end-of-data indicator is sent from a target to an initiator.

```
        Byte 10
    b0  (C) set when status is sent over the control channel.  In
        this case, the Command Status field is gnored.
    b1  (D) set when status is sent over this data channel.
    b2-7 not used (should be set to 0).
```

Bits b0 and b1 are mutually exclusive. It is permissable to not set any flag, in which case the Command Status field is ignored.

For an end-of-data indicator sent from an initiator to a target (as for a WRITE operation), the Status Flags and Command Status fields should be set to zero.

3.8.5. Command Status

3.9. Text Command

The Text Command is provided to allow the exchange of information and for future extensions. It permits the initiator to inform a target of its capabilities or to request some special operations.

```
Byte /   0      |       1       |       2       |       3       |
 /              |               |               |               |
|7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|
 +---------------+---------------+---------------+---------------+
 0| Opcode (0x05) | Reserved (0)  | Length                        |
 +---------------+---------------+---------------+---------------+
 4| Initiator Task Tag                                           |
 +---------------+---------------+---------------+---------------+
 8/ Reserved (0)                                                 /
 +/                                                              /
 +---------------+---------------+---------------+---------------+
40/ Text                                                         /
 +/                                                              /
 +---------------+---------------+---------------+---------------+
```

3.9.1. Length

The length, in bytes, of the Text field.

3.9.2. Initiator Task Tag

The initiator assigned identifier for this Text Command.

3.9.3. Text

The initiator sends the target a set of key:value pairs in UTF-8 unicode format. The key and value are separated by a ':' (0x3A) delimiter. Many key:value pairs can be included in the Text block by separating them with nul '\0' (0x00) delimiters. Some basic key:value pairs are described in Appendix B.

The target responds by sending its response back to the initiator. The target and initiator can then perform some advanced operations based on their common capabalities.

Manufacturers may introduce new keys by prefixing them with their (reversed) domain name, for example,

```
com.foo.bar.do_something:0000000000000003
```

Any key that the target does not understand may be ignored without affecting basic function. Once the target has processed all the key:value pairs, it reponds with the Text Response command, listing the parameters that it supports. It is recommended that Text operations that will take a long time should be placed in their own Text command.

If the Text Response does not contain a key that was requested, the initiator must assume that the key was not understood by the target.

3.10. Text Response

The Text Response message contains the responses of the target to the initiator's Text Command. The format of the Text field matches that of the Text Command.

```
Byte /    0      |       1       |       2       |       3       |
   /             |               |               |               |
  |7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|
  +---------------+---------------+---------------+---------------+
 0| Opcode (0x85) | Reserved (0)  | Length                        |
  +---------------+---------------+---------------+---------------+
 4| Initiator Task Tag                                            |
  +---------------+---------------+---------------+---------------+
 8/ Reserved (0)                                                  /
 +/                                                               /
  +---------------+---------------+---------------+---------------+
40/ Text Response                                                 /
 +/                                                               /
  +---------------+---------------+---------------+---------------+
```

3.10.1. Length

The length, in bytes, of the Text Response field.

3.10.2. Initiator Task Tag

The Initiator Task Tag matches the tag used in the initial Text Command and is used by the initiator to connect Text Commands with the appropriate Text Responses.

3.10.3. Text Response

The Text Response field contains responses in the same key:value format as the Text Command. Appendix B lists some basic Text Commands and their Responses.

3.11. Login Command

After establishing a TCP connection between an initiator and a target, the initiator should issue a Login Command to gain further access to the target's resources.

```
Byte /    0      |      1      |      2      |      3      |
   /             |             |             |             |
  |7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|
  +---------------+---------------+---------------+---------------+
 0| Opcode (0x03) | Reserved (0)  | Length                        |
  +---------------+---------------+---------------+---------------+
 4| Initiator Task Tag                                            |
  +---------------+---------------+---------------+---------------+
 4/ Reserved (0)                                                  /
 +/                                                               /
  +---------------+---------------+---------------+---------------+
40/ Login Parameters in Text Command Format                       /
 +/                                                               /
  +---------------+---------------+---------------+---------------+
```

The initiator may provide some basic parameters in order to enable the target to determine if the initiator may in fact use the target's resources.

The format of the parameters is as specified for the Text Command.

Targets may require keys to indicate the Domain Name of the initiator and the target, and perhaps also an Authenticator key.

The initiator may also provide additional parameters to the target in Text Command format, if the initiator so desires.

Keys and their explanations are listed in Appendix B. An example of the parameters passed is:

```
Initiator:me@my.org
Target:diskfarm.your.org
Authenticator:open-sesame
```

3.12. Login Response

The target responds to the Login Command with a Login Response. It is sufficient for the target to respond with a Status indicating that the Login is accepted.

In fact, the target may completely ignore the parameters that were sent to it and may provide service to any initiator that connects to it. The target may also send back parameters to the initiator in Text Command format, if the target so desires.

In particular, the target may want to provide its Authenticator key, so that the initiator can be sure that it is in fact talking with the correct target.

The initiator can request that the target provide the Authenticator parameter by specifying the SendAuthenticator:yes key:value pair.

```
Byte /    0      |       1       |       2       |       3       |
   /             |               |               |               |
  |7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|
  +---------------+---------------+---------------+---------------+
 0| Opcode (0x83) | Reserved (0)  | Length                        |
  +---------------+---------------+---------------+---------------+
 4| Initiator Task Tag                                            |
  +---------------+---------------+---------------+---------------+
 8/ Reserved (0)                                                  /
 +/                                                               /
  +---------------+---------------+---------------+---------------+
24| Status        | Reserved (0)                                  |
  +---------------+---------------+---------------+---------------+
28/ Reserved (0)                                                  /
 +/                                                               /
  +---------------+---------------+---------------+---------------+
40/ Login Parameters in Text Command Format                       /
 +/                                                               /
  +---------------+---------------+---------------+---------------+
```

The format of the Login Response is the same as the Text Response, with the addition of one field.

3.12.1. Status

The Status returned in a Login Response is one of the following:
  0 accept login (will now accept SCSI commands)
  1 reject login
  2 additional authentication required
In the case that the Status is "accept login" the initiator may

proceed to issue SCSI commands. In the case that the Status is "reject login" the target will immediately close down its end of the TCP conneciton.

In the case that the Status is "additional authentication required" the initiator must provide additional authentication information by issuing the Text Command with the appropriate key:value pairs. (This may be required if the authentication method is based on a challenge/response algorithm.) Upon receipt of the necessary authentication, the target will issue a Login Response with the "accept login" Status. SCSI Commands will not be accepted until the target provides a Login Response with the "accept login" Status.

3.13. Open Data Connections Command

```
Byte /    0        |        1        |        2        |        3        |
   /               |                 |                 |                 |
  |7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|
  +---------------+---------------+---------------+---------------+
 0| Opcode (0x04) | Reserved (0)  | Length (0)                    |
  +---------------+---------------+---------------+---------------+
 4| Initiator Task Tag                                            |
  +---------------+---------------+---------------+---------------+
 8/ Reserved (0)                                                  /
 +/                                                               /
  +---------------+---------------+---------------+---------------+
20| Number of Data Connections Requested                          |
  +---------------+---------------+---------------+---------------+
24/ Reserved (0)                                                  /
 +/                                                               /
  +---------------+---------------+---------------+---------------+
40
```

The initiator requests that some number of data connections (TCP channels) be opened. These may be opened on any network address belonging to the initiator.

3.13.1. Initiator Task Tag

An initiator assigned identification number for this operation.

3.14. Open Data Connections Response

The target responds with the number of data connections it is willing to grant in response to the Open Data Connections Command.

```
Byte /   0      |      1      |      2      |      3      |
   /            |             |             |             |
  |7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|
  +---------------+---------------+---------------+---------------+
 0| Opcode (0x84) | Reserved (0)  | Length                        |
  +---------------+---------------+---------------+---------------+
 4| Initiator Task Tag                                            |
  +---------------+---------------+---------------+---------------+
 8/ Reserved (0)                                                  /
 +/                                                               /
  +---------------+---------------+---------------+---------------+
20| Number of Data Connections Granted                            |
  +---------------+---------------+---------------+---------------+
24| Reserved (0)                                                  |
 +/                                                               /
  +---------------+---------------+---------------+---------------+
40| Cookie                                                        |
 +/                                                               /
  +---------------+---------------+---------------+---------------+
72/ Target Address Names (and counts) for Data Connections        /
 +/                                                               /
  +---------------+---------------+---------------+---------------+
```

3.14.1. Cookie

The target provides the initiator with a 32 byte entity (referred to here as a "cookie"), which the initiator will later use to identify itself to the target. The initiator will open data connections and will provide the cookie back to the target, so that the target can map a newly established data connection to an existing control connection. Upon establishing the data connection, the initiator must send the cookie as the first bytes of data sent through the data connection.

Cookies might be valid for only a short period of time (as determined by the target). If a long time passes (as determined by the target) after the Open Data Connections Response and a data connection has not been established, the target may invalidate the cookie. If an initiator tries to use a cookie that has expired, the target immediately closes the data connection. The target may send an Asynchronous Event to the initiator (over the corresponding control connection) to inform it that an expired cookie was used.

3.14.2. Target Address Names

The target may have several addresses through which it allows data connections. The target may also allow a different number of connetions to each of those addresses, based, perhaps, on the bandwidth of the connections that those addresses represent. These are specified in the Target Address Names field in Text Command format. For each target address to be used for data connections, the target specifies the Name of the address and number of connections allowed on that address with a colon ':' separating Address Name from the number of connetions. The target may also specify the number of connections without specifying the address, thereby implying use of the same Domain Name that was used for the control connection over which the Open Data Connections Command arrived. In this case, no Address Name preceeds the colon; only the number of connections follows the colon in Text format. Entries are separated by nul '\0' (0x00). For example:

```
name1.trg.org:3
:2
name3.trg.org/disk3:4
```

This example indicates that 3 data channels may be opened at the address named "name1.trg.org" and 4 data channels may be opened at the address named "name3.trg.org/disk3." In addition, 2 data connections may be made at the address that was used for the control connection.

Note that the sum of the individually specified numbers (per-target address) for data connections as they appear in the Text format may be more than the number specified in the "Number of Data Connections Granted" field. The target simply specifies where it it convenient for the target to receive data connections. Upon establishing the granted number of data connections, the target may disallow any further data connections from the initiator.

3.14.3. Length

The Length parameter contains the length of the cookie plus the total length of the Text containing address and connection information.

3.15. Ping Command

```
Byte /   0     |     1     |     2     |     3     |
 /          |          |          |          |
|7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|
+---------------+---------------+---------------+---------------+
0| Opcode (0x0)  | Reserved (0)  | Length                        |
+---------------+---------------+---------------+---------------+
4| Initiator Task Tag                                          |
+---------------+---------------+---------------+---------------+
8/ Reserved (0)                                               /
+/                                                           /
+---------------+---------------+---------------+---------------+
40/ Ping Data (optional)                                      /
+/                                                           /
+---------------+---------------+---------------+---------------+
```

The Ping Command can be used to verify that a control connection is still active. This command may be useful in the case where an initiator has been waiting a long time for the response to some command, and the initiator suspects that there is some problem with the connection.

When a target receives the Ping Command, it should respond with a Ping Response, duplicating the data that was provided in the Ping Command, if present.

If the initiator does not receive the Ping Response within some period of time (determined by the initiator), or if the data returned by the Ping Response is different from the data that was in the Ping Command, the initiator may conclude that there is a problem with the control connection. The initiator will then close the control connection and may try to establish a new control connection.

3.15.1. Length

The length of the optional Ping Data.

3.15.2. Initiator Task Tag

An initiator assigned identifier for the operation.

3.15.3. Ping Data

Binary data that will be reflected in the Ping Response.

3.16. Ping Response

```
Byte /    0     |      1      |      2      |      3      |
   /            |             |             |             |
  |7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|
  +---------------+---------------+---------------+---------------+
 0| Opcode (0x80) | Reserved (0)  | Length                        |
  +---------------+---------------+---------------+---------------+
 4| Initiator Task Tag                                            |
  +---------------+---------------+---------------+---------------+
 8/ Reserved (0)                                                  /
 +/                                                               /
  +---------------+---------------+---------------+---------------+
40/ Return Ping Data                                              /
 +/                                                               /
  +---------------+---------------+---------------+---------------+
```

When a target receives the Ping Command, it should respond with a Ping Response, duplicating the data and Initiator Task Tag that was provided in the Ping Command, if present.

3.17. Third Party Commands

There are some third-party SCSI commands, such as COPY and EXTENDED COPY, that require one target (Target A) to act as an initiator to other targets (e.g., Target B). Some such commands can be extended in a straightforward way to accommodate new forms of addressing, and this should be done to address targets using SCSI/TCP. These extensions are not straightforward for all commands, and they may not be able to encompass the full name space and authentication information needed for SCSI/TCP in some contexts. Thus SCSI/TCP also provides a facility for assigning local short-form aliases to full addressing/authorization information for targets, and the aliases can be used in the SCSI commands and parameter data. The alias information is specified as Text following the header of the SCSI command specifying the third-party command. The header will thus appear as follows:

```
Byte /   0     |     1     |     2     |     3     |
 /           |           |           |           |
 |7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|
 +---------------+---------------+---------------+---------------+
 0| Opcode (0x01) | Reserved (0)  | Length (!= 0)                 |
 +---------------+---------------+---------------+---------------+
 4| Initiator Task Tag                                            |
 +---------------+---------------+---------------+---------------+
 8| Logical Unit Number (LUN)                                     |
 +                                                               +
12|                                                               |
 +---------------+---------------+---------------+---------------+
16| Flags         |Task Attributes| Reserved (0)                  |
 +---------------+---------------+---------------+---------------+
20| Expected Data Transfer Length                                 |
 +---------------+---------------+---------------+---------------+
24| SCSI Command Block (SCB)                                      |
 +                                                               +
28|                                                               |
 +                                                               +
32|                                                               |
 +                                                               +
36|                                                               |
 +---------------+---------------+---------------+---------------+
40/ Data/Parameters for SCB                                       /
 +/ Parameters needed for Target B                                /
 /                                                               /
 +---------------+---------------+---------------+---------------+
```

The Length field will not be zero. Rather, it will contain the length of the alias information which may include the name of

Target B and an Authentication key in Text Command format. An example of the data for this command might be:

```
LocalName:TargetB
FullName:disk2.sj.foo.com
OriginalAuthenticator:open-sesame
```

If the SCSI command requires data/parameters beyond the 16-byte SCB, the target can figure this out by examining the particular SCSI command and the other contents of the SCB. Any data (as specified by the Length parameter) beyond what is needed for the specific SCSI command are parameters in Text Command format needed to connect to Target B.

Upon receiving a third-party command, Target A will perform login operations with the identified targets. In effect, Target A will become an initiator to Target B. Among the parameters provided to Target B, Target A may specify the authentication information from the initiator. The Text provided by Target A when it performs the Login command to Target B may contain the keys Target (referring to Target B) and Initiator (referring to Target A), and it may also contain the keys Authenticator (of Target A), OriginalInitiator and OriginalAuthenticator (referring to authenticator of the original initiator).

3.18. Opcode Not Understood

```
Byte /    0    |    1    |    2    |    3    |
 /            |           |          |          |
 |7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|
 +---------------+---------------+---------------+---------------+
 0| Opcode (0x88) | Reserved (0)  | Length                        |
 +---------------+---------------+---------------+---------------+
 4/ Reserved (0)                                                 /
 +/                                                             /
 +---------------+---------------+---------------+---------------+
40/ Header of Bad Message                                        /
 +/                                                             /
 +---------------+---------------+---------------+---------------+
80
```

It may happen that a target receives a message with an Opcode that it doesn't recognize. This may occur because of a new version of the protocol that defines a new Opcode, or because of some corruption of a message header.

The target returns the header of the message with the unrecognized opcode as the data of the response.

4. Error Handling SCSI/TCP

The following errors might be detected by the initiator and targets:

- TCP connection termination with outstanding operations pending
- Operation timed out
- Illegal field format---could be an implementation error or a data stream synchronization error.

Initiators that detect one of the above errors will take the following actions:

- Reset the TCP connection.
- Terminate all outstanding requests on behalf of the LUNs on the target with the error.
- Reestablish the session by:
    - Opening a new TCP connection.
    - Performing the login process.

Note: After the initiator resets its TCP connection, the target connection may remain active. If a new TCP connection is opened by the initiator, it can lead to two active connections at the target side for the same initiator. The target is not aware of the cleanup actions taken by the initiator. To avoid this situation, the target has to check on every Login to see whether or not it established an already existing session. If the session already exists, the target performs cleanup actions (described in the following paragraph) for the old session before performing the new Login.

A target that detects one of the above errors will take the following actions:

- Reset the TCP connection.
- Abort all Tasks in the task set for the corresponding initiator.

5. Notes to Implementors

This protocol was designed to take advantage of the Remote DMA TCP options [RDMA], although it can still operate effectively without this TCP extension. This section notes some of the performance considerations of the SCSI/TCP protocol.

5.1. Small TCP Segments

It is recommended that TCP segments be limited in size to no more than 8K bytes. One reason is to ensure that segments won't get broken into smaller packets, thereby possibly breaking the assumptions for RDMA and the information in the RDMA header. Another reason we recommend small segments is to allow a stronger type of checksum, possibly utilizing CRC, which is practical only for smaller segments.

5.2. Multiple Network Adapters

The SCSI/TCP protocol assumes that the Task Tags will also serve as RDMA tags. The SCSI/TCP protocol allows multiple data connections, not all of which need go over the same network adapter. If multiple network connections are to be utilized with RDMA, the SCSI/TCP protocol requires that the Task Tag (RDMA Tag) contain sufficient information (and sufficient support from the various network adapters) to allow data to arrive on any of the data connections, even if they are not all through the same network adapter.

5.3. Autosense

Autosense refers to the automatic return of sense data to the initiator in case a command did not complete successfully. If autosense is turned off, the initiator must explicitly request that sense data be sent to it after some command has completed with a CHECK CONDITION status. The default for SCSI/TCP is to work with Autosense enabled.

Note that even if a SCSI target/LUN does not support Autosense, it may still be possible for SCSI/TCP to work with Autosense. This can be accomplished as follows. Whenever a CHECK CONDITION status is about to be returned, the SCSI/TCP component on the target immediately queries the target/LUN for the sense data. SCSI/TCP can then return the sense data to the initiator together with the CHECK CONDITION status. It is not necessary for SCSI/TCP to wait for the initiator to explicitly request the sense data; the target SCSI/TCP code can perform this operation automatically, even for devices/LUNs that do not ordinarily provide automatic sense data.

5.4. TCP RDMA option

The TCP RDMA option [RDMA] is an annotation on individual TCP segments that can reduce the number of copies necessary at the receiver. The RDMA option succinctly describes the portion of a TCP payload that holds bulk data.

In SCSI/TCP, all bulk data transfers occur over the data connections. Thus, RDMA options will only appear on the data connection.

To disambiguate between multiple transfers on a TCP connection, a 48-bit RDMA ID (RID) appears in the TCP option. In the case of SCSI/TCP, the upper 16 bits of the RID are zero and the lower bits are the same as the Transfer Tag on the data message.

In the case of an initiator to target data phase without ready-to-transfer (RTT), the unsolicited bit should be set in the RDMA option.

5.5. Data Connection Options

Some targets may want to inform (or negotiate with) an initiator concerning some parameters related to bandwidth, Quality of Service, or some other available features on its various network connections. These are exchanged between the initiator and the target using Text Commands and Responses. These should be exchanged after a successful login, but before any data connections are established.

6. Security Considerations

6.1. Data Integrity

We assume that end-to-end data integrity can be assured by TCP, by adding a more powerful checksum option whenever this is considered important, or replacing the checksum by a weaker one (or even "nullifying it") for applications in which data integrity is not important and recovery from data errors could be harmful (e.g., audio or video distribution streams).

6.2. Login Process

In some environments, a target will not be interested in authenticating the initiator. In this case, the target can simply ignore some or all of the parameters sent in a Login Command, and the target can simply reply with a basic Login Response indicating a successful login.

Some targets may want to perform some kind of authentication. The Authenticator key is defined for this purpose. Various authentication schemes can be used, including encrypted passwords and trusted certificate authorities.

Once the initiator and target are confident of the identity of the attached party, the established control channel is considered secure. The initiator then proceeds to request (over the secure control channel) the allocation of data channels. The target provides the initiator with a cookie that must be used when establishing a data channel. This enables the target to match a data channel to its corresponding control channel. The target may set a time limit to the validity of a cookie that it has provided for data connections.

It is anticipated that most target devices will not bother with all of the possible checks, but the protocol provides sufficient means to perform the checks, if required by the target.

6.3. IANA Considerations

There will be a well known port for SCSI/TCP control connections and a well known port for SCSI/TCP data connections. These well known ports will have to be registered with IANA.

A checksum type will also have to be registered with IANA.

7. Authors' Addresses

Julian Satran
Kalman Meth
Meir Toledano
IBM, Haifa Research Lab
MATAM - Advanced Technology Center
Haifa 31905, Israel
Phone +972 4 829 6211
Email: satran@il.ibm.com meth@il.ibm.com
toledano@il.ibm.com

Daniel F. Smith
Prasenjit Sarkar
Carlos Fuente
IBM Almaden Research Center
650 Harry Road
San Jose, CA 95120-6099, USA
Phone: +1 408 927 2072
Email: dfsmith@almaden.ibm.com psarkar@almaden.ibm.com
carlos_fuente@us.ibm.com

Costa Sapuntzakis
Cisco Systems, Inc.
170 W. Tasman Drive
San Jose, CA 95134, USA
Phone: +1 408 525 5497
Email: csapuntz@cisco.com

Efri Zeidner
SanGate
Israel

Comments may be sent to Julian Satran, Daniel Smith, Costa Sapuntzakis, or Kalman Meth.

8. References and Bibliography

[RDMA] Internet Draft: TCP RDMA option (work in progress)
[SAM2] ANSI X3.270-1998, SCSI-3 Architecture Model (SAM-2)
[TLS] The TLS Protocol, RFC 2246, T. Dierks et al.

[ALTC] Internet Draft: Alternative checksums (work in progress)
[CAM] ANSI X3.232-199X, Common Access Method-3 (Cam-3)
[CRC] ISO 3309, High-Level Data Link Control (CRC 32)
[RFC793] Transmission Control Protocol, RFC 793
[RFC1122] Requirements for Internet Hosts -- Communication Layer, RFC 1122, R. Braden (editor)
[SBC] ANSI X3.306-199X, SCSI-3 Block Commands (SBC)
[SCSI2] ANSI X3.131-1994, SCSI-2
[SPC] ANSI X3.301-199X, SCSI-3 Primary Commands (SPC)

9. Appendix A - Examples

9.1. Read operation example

Initiator Function	Message Type	Target Function
Command request (read)	SCSI Command (READ)>>> (on control channel)	
		Prepare Data Transfer
Receive Data	<<< SCSI Data (on data channel)	Send Data
Receive Data	<<< SCSI Data (on data channel)	Send Data
Receive Data	<<< SCSI Data (on data channel)	Send Data
	<<< SCSI Response (on control channel)	Send Status and Sense
Command Complete		

9.2. Write operation example

Initiator Function	Message Type	Target Function
Command request (write)	SCSI Command (WRITE)>>> (on control channel)	Receive command and queue it
		Process old commands
	<<< RTT (on control channel)	Ready to process WRITE command
Send Data	SCSI Data >>> (on data channel)	Receive Data
Send Data	SCSI Data >>> (on data channel)	Receive Data
	<<< RTT (on control channel)	
Send Data	SCSI Data >>> (on data channel)	Receive Data
	<<< SCSI Response (on control channel)	Send Status and Sense
Command Complete		

10. Appendix B - Login/Text keys

10.1. Target

Target:domainname[/modifier]

Examples:

```
Target:disk-array.sj-bldg-h.cisco.com
Target:disk-array.sj-bldg-h.cisco.com/disk3
```

This key is provided by the initiator of the TCP connection to the remote endpoint. The Target key specifies the domain name of the target, since that information is not available from the TCP layer. The target is not required to support this key.

The initiator should send this key in the first login message. The Target key might be used by application layer proxies to learn the intended endpoint of the conversation.

10.2. Initiator

Initiator:[principal@]domainname

Examples:

```
Initiator:sample.foobar.org
Initiator:fred@sample.foobar.org
Initiator:fred@
Initiator:
```

The Initiator key enables the initiator to identify itself to the remote endpoint. A zero-length principal is valid and indicates that the initiator has no specific principal to communicate to the target. The domain name should be that of the initiator. A zero-length domain name is interpreted as "other side of TCP connection". The target may silently ignore this key if it does not support it.

For more security, a certificate-based protocol [TLS] may be used on the channel and take precedence over this protocol.

10.3. Authenticator

Authenticator:<UTF8-String>

Examples:

```
Authenticator:open-sesame
```

The authenticator is a secret that the initiator uses to gain access to the target's LUNs.

10.4. SendAuthenticator

SendAuthenticator:yes Response: Authenticator:<UTF8-String>

Examples:

```
SendAuthenticator:yes
-> Authenticator:alakazam
```

The SendAuthenticator key is used to request from the party on the other side of the TCP connection to send its Authenticator. SCSI/TCP devices may refuse to grant access until proper authentication has been performed by the parties involved.

10.5. AllowNoRTT

AllowNoRTT:<yes|no> Response: AllowNoRTT:<yes|no>

Examples:

```
AllowNoRTT:yes
-> AllowNoRTT:yes
```

The AllowNoRTT key is used to allow an initiator to send data to a target without the target having sent an RTT to the initiator. The default action is that RTT is required, unless both the initiator and the target send this key-pair attribute specifying AllowNoRTT:yes. Once AllowNoRTT has been set to 'yes', it cannot be set back to 'no'.

10.6. OriginalInitiator

OriginalInitiator:principal@domainname

Examples:

OriginalInitiator:fred@sample.foobar.org

The OriginalInitiator key is used to perform a proxy login from one target to another target in order to perform a third-party operation (like COPY) for some initiator. The first target acts as the initiator for the second target, but it must provide the authorization information of the original initiator.

10.7. Target2

Target2:domainname[/modifier]

Examples:

Target2:sample.foobar.org
Target2:sample.foobar.org/disk2

The target2 key is used in a third-party SCSI command (like COPY) between targets that do not lie on the same SCSI fabric. The initiator must specify the name of the distant target to the original target, so that the original target can Login to the distant target and then perform the third-party command.

Expires 14 August 2000